Episode 6.05 – Don't Cares, the Logical Kind

Welcome to the Geek Author series on Computer Organization and Design Fundamentals. I'm David Tarnoff, and in this series, we are working our way through the topics of Computer Organization, Computer Architecture, Digital Design, and Embedded System Design. If you're interested in the inner workings of a computer, then you're in the right place. The only background you'll need for this series is an understanding of integer math, and if possible, a little experience with a programming language such as Java. And one more thing. Due to the computational nature of this episode, you might want to visit the transcript page found at intermation.com to download the episode worksheet.

Assume you've been invited to play poker with some friends. The game is five card draw and jacks are wild. What does it mean that jacks are wild? It means that you can change any jacks dealt to you to whatever suit or rank you need in order to get the best possible hand. At one point during the game, you find yourself holding the following five cards: ace of spades, ace of clubs, ace of hearts, jack of diamonds, and a two of diamonds. Wow, that's some hand. Three aces are pretty good, but remember, jacks are wild and you can change the jack of diamonds to anything you want. Can we make this hand even better? Changing the jack to a three of diamonds would be, well, just wrong. It makes the hand no better. But we have some other options. Changing it to a two of diamonds would give you a full house: three of a kind and a pair. That's an improvement. Changing it to an ace, however, would give you four of a kind, an even better hand beatable by only a royal flush or a straight flush. Actually, five of a kind could beat it too, a hand possible only with wild cards.

It turns out that we have wild cards when it comes to truth tables too. These signal values, referred to as "don't cares" (yes, that's really what we call them) are represented in the truth table with a capital X. They could be found in the input columns or the output column. As an output, this symbol means that for a particular set of input values, we don't care what the circuit output is. It could be that for a certain combination of inputs, the output of our circuit doesn't matter. It could also be that a certain combination of inputs isn't possible for this circuit, in which case, we also don't care what the circuit output is.

Our first example may seem a little odd, but it has a bunch of places where the output is a "don't care". Let's say that we want to create a circuit that takes as its input a four-bit binary value meant to represent a single decimal digit in binary coded decimal or BCD. The output of this circuit needs to be one if the BCD decimal value is divisible by three. The decimal digits divisible by three are zero, three, six, and nine, right? That means our truth table will have ones in the output column for the rows 0-0-0-0, 0-0-1-1, 0-1-1-0, and 1-0-0-1. There will be a zero output for the rows representing one, two, four, five, seven, and eight. But what about the rows past nine? These rows do not represent single digit decimal values. Since these patterns are not BCD values, the output doesn't matter to us. Therefore, the output column should have an X for the rows 1-0-1-0, 1-0-1-1, 1-1-0-0, 1-1-0-1, 1-1-1-0, and 1-1-1-1. This gives us the following values as we read down the output column of the truth table from the top to the bottom: 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, X, X, X, X, X, X.

In other cases, certain combinations of inputs shouldn't be possible. For example, picture a thermostat that has two separate input signals. One of them, which we will call A, is used to identify when the temperature in a room has gone "above" a maximum temperature, and the air conditioning should be turned on. A second binary signal, which we will call B, is used to identify when the temperature in a

room has fallen “below” a minimum temperature indicating that the heat should be turned on. If either the heat or the air conditioning are turned on, a blower fan should be turned on too. So what does the truth table look like for the logic controlling the blower fan? The truth table has two inputs, A and B, and therefore, has four rows: 0-0, 0-1, 1-0, and 1-1. For 0-0, the fan should be off. Neither the air conditioning nor the heat are on, so no fan is needed. For 0-1, the fan should be on because the heat is on. For 1-0, the fan should be on because now the air conditioning is on. But what about 1-1? Well, this condition should never occur, so as far as the logic is concerned, the output is a “don’t care”. This gives us the values as we read the output column from top to bottom as 0, 1, 1, X. In reality, this might not be the best example as a situation where both the heat and the air-conditioning are on simultaneously may indicate a fault, and we may want to turn on the fan in order to keep heat from building up in the furnace.

If a truth table contains one or more “don’t care” elements, those “don’t cares” are transferred to the corresponding cells of the Karnaugh map. The question is, do we include a “don’t care” in a rectangle or not? Well, just like the poker hand, you do what best suits the situation. To see how this is done, let’s transfer our ealier example of checking a decimal digit for divisibility by three to a four-input Karnaugh map. We will identify the binary digits from left to right as A, B, C, and then D. There are four cells in the Karnaugh map that have ones in them: the leftmost column of top row, 0-0-0-0, the third column of the top row, 0-0-1-1, the rightmost column of the second row, 0-1-1-0, and the second column of the bottom row, 1-0-0-1. There are six cells that have the X’s representing the don’t cares: all four cells in the third row representing the truth table rows 1-1-0-0, 1-1-0-1, 1-1-1-1, and 1-1-1-0. There are also two don’t cares in the two right cells of the bottom row for the truth table rows 1-0-1-1 and 1-0-1-0.
If we hadn’t used don’t cares and instead had placed zeros in the six cells representing the non-decimal nibbles, all four of the cells containing ones in this Karnaugh map would have been isolated, surrounded by zeros on all four sides. The resulting products would have been A-bar AND B-bar AND C-bar AND D-bar for the top left cell, A-bar AND B-bar AND C AND D for the third cell of the top row, A-bar AND B AND C AND D-bar for the rightmost cell of the second row, and A AND B-bar AND C-bar AND D for the second cell of the bottom row.

With the don’t care wild cards, however, three of the four cells can be expanded into larger rectangles making simpler products or implicants. Remember that the don’t cares can be set to either zero or one, so the cells they occupy in the Karnaugh map can be included or not included in the rectangles. Only include a don’t care in a rectangle if it makes the rectangle bigger. Never add a rectangle just to include a don’t care.

First, the third cell in the top row representing 0-0-1-1 could be paired with the don’t care cell in the third column on the bottom row. This rectangle now crosses both values of A, so A drops out and the term A-bar AND B-bar AND C AND D becomes B-bar AND C AND D. Next, the cell in the second column of the bottom row can be doubled by including the cell above it which contains an X. This new rectangle crosses both values of B, so B drops out. This rectangle can be doubled in size again to four cells by including both the cells to the right containing X’s. This crosses both values of C, so the C term drops out. This reduces the product A AND B-bar AND C-bar AND D to just A AND D. Lastly, the cell in the rightmost column of the second row can be paired with the cell below it which contains an X. This crosses both values of A, and the term A-bar AND B AND C AND D-bar becomes B AND C AND D-bar. The cell in the top left corner, however, is the only one that cannot be simplified with the addition of the don’t cares. Through the use of these don’t care wildcards, our most simplified boolean sum-of-products expression is now A·B·C·D + B·C·D + A·D + B·C·D (read A-bar AND B-bar AND C-bar AND D-bar OR-ed with B-bar AND C AND D OR-ed with A AND D OR-ed with B AND C AND D-bar).

All of the cells containing don’t cares that are now covered by rectangles are considered to contain ones. Two cells containing don’t cares, the one in the leftmost column of the third row and the one in the bottom right corner, are not covered by rectangles and are considered to contain zeros. The final circuit will output these values due to the simplified expression. Later in this series, we will examine some cases where we will want to see if those values that were assigned to “don’t cares” by being included or not included in a rectangle could cause a problem.

In our next episode, will show how don’t cares can be used to simplify our truth tables. For episode transcripts, worksheets, links, or other podcast notes, please visit us at intermation.com where you will also find links to our Instagram, Twitter, Facebook, and Pinterest pages. Until the next episode, remember that while the scope of what makes a computer is immense, it’s all just ones and zeros.